ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Dipartimento di Ingegneria Industriale

Corso di Laurea in Ingegneria Aerospaziale

# Aspetti di modellazione e simulazione di un veicolo lanciatore in scala

**Relatore:**
**Chiar.mo Prof.**
**Fabrizio Giulietti**

**Presentata da:**
**Carlo Gargiulo**

---

**Sessione Marzo 2025**
**Anno Accademico 2023/2024**

*Alla mia famiglia*
*che mi ha guidato fino a qui*

**Sommario**

In questo testo si realizza un modello per propulsione, dinamica e cinematica di un razzo ad acqua e lo si confronta con i dati raccolti sperimentalmente con un'unità inerziale, viene realizzato un ambiente per raccogliere e utilizzare i dati estratti dall'unità inerziale.

Nella fase di simulazione i modelli vengono scritti in Matlab-Simulink sotto forma di equazioni differenziali lineari di primo ordine per poi essere integrate nel tempo per ottenere i valori istante per istante.
Nella fase di raccolta dati, essi vengono estratti da scheda SD, decodificati e importati in ambiente Matlab per l'analisi e realizzazione dei grafici.
I due risultati vengono confrontati in forma grafica nella traiettoria, velocità e accelerazione, dimostrando una discreta accuratezza del modello creato.

*Parole chiave: razzo ad acqua, simulazione, unita inerziale, Simulink, equazioni razzo, dinamica lanciatore, lanciatore in scala, razzo ad acqua propulsione*

# Indice

# 1 Introduzione

Un razzo ad acqua è un razzo che sfrutta come propulsione un getto d'acqua messo in moto da aria compressa, solitamente composto da una bottiglia di plastica, delle alette e un sistema per pressurizzare e rilasciare il razzo stesso.
Noto come strumento didattico per dimostrare l'applicazione della terza legge della dinamica e apprezzato per la sua semplicità costruttiva e operativa, con un basso rischio di incidenti e la lieve entità di essi.

Diversi studi esistono già in letteratura sul moto di questo oggetto e di come sia possibile modellare il getto d'acqua che muove il razzo.
Shaviv[3] fornisce un semplice modello ottimo per la prima iterazione del simulatore di volo del razzo, Utsav e Rafat[4] esplicano un modello più complesso che permetta di tenere conto di un numero maggiore di fattori nella simulazione del comportamento di gas e acqua nella bottiglia.
Infine Feeley e Speyer[2] forniscono un set di equazioni per descrivere il moto del razzo, grazie alle quali è stato possibile creare il simulatore di volo in Simulink.

In questa tesi si affronterà la realizzazione di un modello per la propulsione e il volo del razzo ad acqua, il confronto del modello con dei dati sperimentali ed inoltre la raccolta e il processo dei dati stessi; offrendo un approccio completo all'attività del lancio del razzo ad acqua affrontando il tema su tutta la linea del problema.
In particolare i primi capitoli affronteranno il modello propulsivo e costitutivo, subito dopo verrà discusso l'utilizzo dell'unità inerziale con limitazioni e possibilità che essa offre ed infine verrà trattata la simulazione numerica di dinamica e cinematica con i modelli discussi precedentemente.
Infine abbiamo una rapida discussione sui risultati raggiunti, sulle limitazioni riscontrate e sulle possibili soluzioni.

Questa attività ha permesso di applicare concetti da diverse materie affrontate durante il corso di studi, quali: Analisi numerica, Meccanica del volo, Propulsione ed Elaborazione dati, oltre a diverse conoscenze apprese durante il percorso universitario.
Per il completamento di questo progetto sono stati utilizzati i programmi Matlab, Matlab-Simulink, Visual Studio Code, Witmotion e Overleaf.

# 2 Modello Propulsivo

Per poter simulare correttamente il moto del razzo si è partiti dalla realizzazione di un modello per il sistema propulsivo.
Partendo dalla legge di conservazione della quantità di moto sappiamo che la derivata della quantità di moto rispetto al tempo è uguale alla risultante delle forze esterne.

$$\frac{dq}{dt} = \frac{d}{dt}(mv) = \frac{dm}{dt}v + m\frac{dv}{dt} = \sum F_e$$

Per la nostra trattazione considereremo la derivata della massa trascurabile. La risultante delle forze esterne invece può essere espressa come:

$$\sum F_e = (p_i - p_e)A_e + \dot{m}u_e = S$$

Dove $p_i, p_e, A_e$ rappresentano rispettivamente la pressione interna, esterna (ambiente) e l'area dell'ugello, $\dot{m}$ è la portata in massa che passa attraverso l'ugello e $u_e$ la velocità di uscita della stessa.
Opereremo qui un'altra semplificazione ignorando il termine di differenza di pressione, ottenendo come formulazione della spinta $S$:

$$S = \dot{m}u_e \tag{2.1}$$

## 2.1 Regimi di Volo

Nella caratterizzazione della spinta del razzo ad acqua possiamo definire quattro regimi, con altrettanti modelli ed equazioni differenti.

1. **Primo regime**, si ha aria compressa nella parte superiore della bottiglia e acqua al di sotto di essa, l'aria si trova ad una pressione più elevata rispetto a quella atmosferica e questa differenza di pressione crea una forza applicata all'acqua sottostante mettendola in moto verso l'ugello. Tuttavia all'interno della bottiglia è ancora presente la spina utilizzata per pressurizzare l'aria dentro la bottiglia, la presenza della stessa limita il flusso e quindi la spinta erogata.

2. **Secondo regime** , la spina è stata espulsa completamente, in questa fase vediamo il massimo della spinta.

3. **Terzo regime**, tutta l'acqua è stata espulsa, all'interno della bottiglia è presente solo aria ancora ad una pressione superiore a quella esterna, che però fornisce ancora della

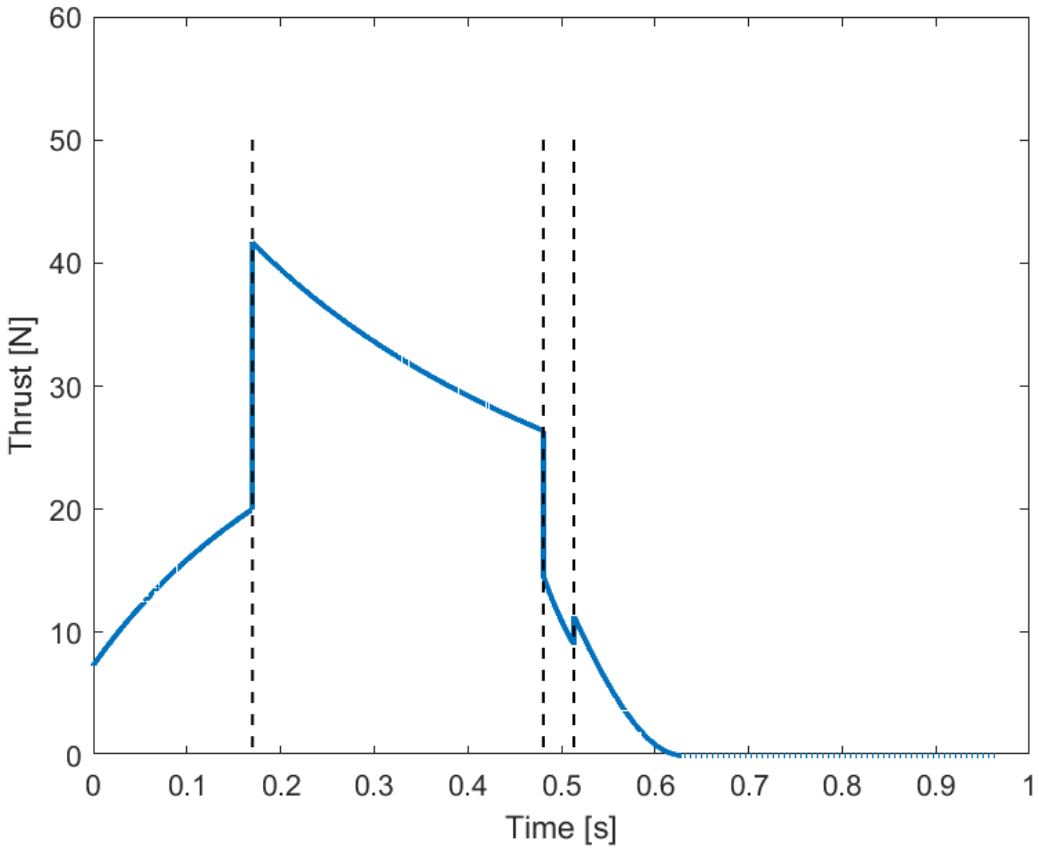



Figura 2.1: 4 Regimi di Volo

spinta. La differenza di pressione è talmente grande che il flusso in gola incontra effetti di sonicità che limitano il flusso dell'uscita dell'aria "strozzandolo".

4. **Quarto regime**, la pressione interna è scesa sotto una soglia critica e il flusso non è più limitato, abbiamo gli ultimi istanti di spinta fino a riportare la pressione interna ed esterna all'equilibrio.

## 2.2 Primo e Secondo Regime

Inizieremo descrivendo il secondo, cioè quello di spinta libera e poi si tenterà di adattarlo al primo regime con delle considerazioni sperimentali. Definiamo delle relazioni tra grandezze che ci permettano quindi di calcolare $\dot{m}$ e $u_e$ date le condizioni iniziali note del razzo [4], quali $p_0$ pressione interna iniziale e $L_0$ pelo libero iniziale.

L'equazione di Bernoulli per trovare un legame tra la velocità all'ugello $u_e$ e pressione interna ed esterna. $g$ è l'accelerazione gravitazionale mentre z è l'altezza della colonna d'acqua nella bottiglia.

$$\frac{{u_e}^2}{2} + \frac{p_e}{\rho_w} = \frac{p}{\rho_w} + gz \tag{2.2}$$

Considerando l'acqua incomprimibile, possiamo trovare la velocità di discesa del pelo libero dell'acqua $u_s$ grazie all'equazione di continuità della massa.

$$u_s = \frac{u_e A_e}{A_s} \tag{2.3}$$

dove $A_e$ è l'area dell'ugello e $A_s$ l'area della bottiglia.

La lunghezza della colonna d'acqua $z$ può essere descritta come:

$$z = L_{pc} - \frac{p_0}{p}(L_{pc} - L_0) \tag{2.4}$$

La portata in massa può essere scritta come:

$$\dot{m} = A_e \rho u_e \tag{2.5}$$

dove $\rho$ è la densità di ciò che sta uscendo dall'ugello, che si tratti di acqua o aria.

Siamo in grado a questo punto di scrivere la velocità all'ugello come:

$$u_e = \sqrt{\frac{2\frac{p-p_e}{\rho_w} + 2g[L_{pc} - \frac{p_0}{p}(L_{pc} - L_0)]}{1 - (\frac{A_e}{A_s})^2}} \tag{2.6}$$

e quindi la spinta come

$$F = \dot{m} u_e = A_e \rho u_e^2 = A_e \rho \frac{2\frac{p-p_e}{\rho_w} + 2g[L_{pc} - \frac{p_0}{p}(L_{pc} - L_0)]}{1 - (\frac{A_e}{A_s})^2}$$

Per modellare il primo regime, cioè quello con la spina inserita che ostruisce la fuoriuscita dell'acqua limitando la spinta e la portata in massa, si è deciso un approccio sperimentale.
Per fare ciò è stato definito un tempo caratteristico per il rilascio del razzo dal supporto di terra $t_{libero} = 0.180s$, definito in base all'analisi degli andamenti dei vari lanci effettuati.

$$S = \dot{m} u_e \left( \frac{1}{4} + \frac{3t}{5t_{libero}} \right) \tag{2.7}$$

$$\dot{m} = \rho A_e u_e \frac{3}{5}$$

quindi

$$S = \rho A_e {u_e}^2 \left( \frac{3}{20} + \frac{9t}{25t_{libero}} \right) \tag{2.8}$$

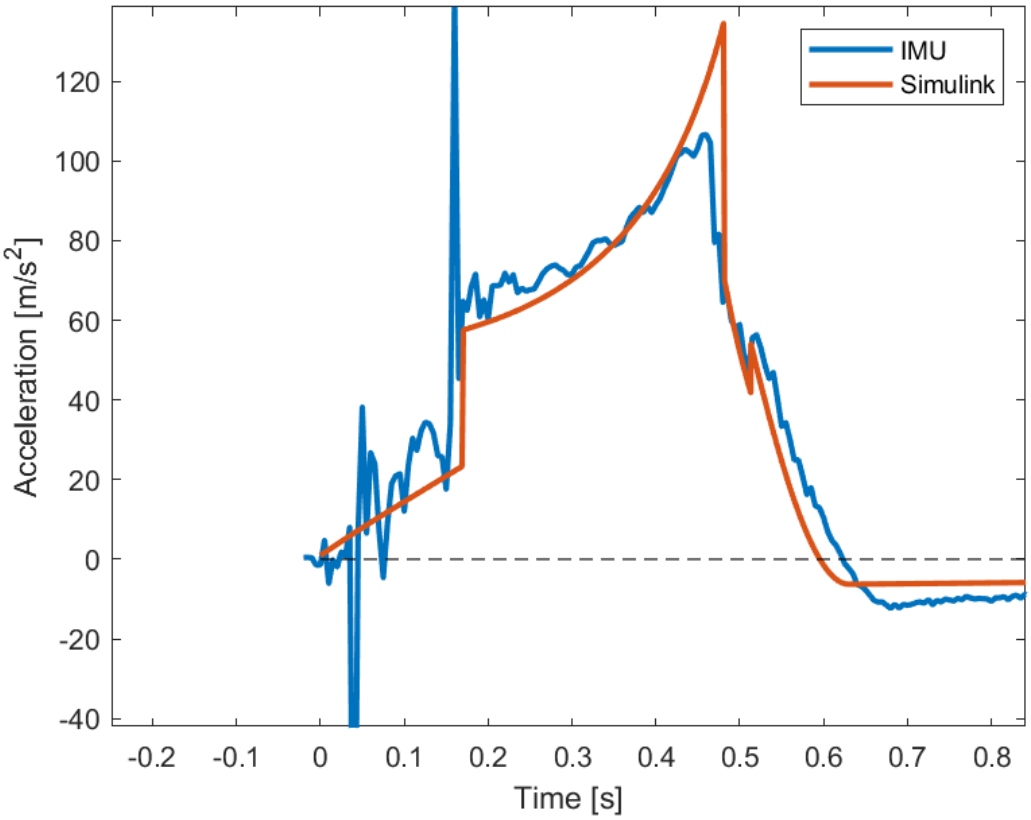



Figura 2.2: Accelerazioni reali e simulate

## 2.3 Terzo Regime

Nel terzo regime, che avviene quando l'acqua è terminata, abbiamo comunque della spinta data dall'uscita dell'aria. Poiché il flusso attraverso l'ugello non è in grado di accelerare oltre una certa soglia per effetti legati alla velocità del suono, la portata in massa risulta costante e limitata.
Le equazioni qui implementate sono quelle standard per un ugello convergente con serbatoio in pressione.

$$\dot{m} = \frac{PA_e}{\sqrt{T}} \sqrt{\frac{\gamma}{R} \left( \frac{2}{\gamma + 1} \right)^{\frac{\gamma+1}{\gamma-1}}} \tag{2.9}$$

La soglia critica si verifica quando il rapporto tra pressione interna ed esterna è superiore a:

$$\frac{p}{p_e} \geq (\frac{2}{\gamma + 1})^{-\frac{\gamma}{\gamma-1}} \tag{2.10}$$

Dove $\gamma$ è il fattore di espansione isoentropica, cioè il rapporto tra il calore specifico a pressione costante $c_p$ ed il calore specifico a volume costante $c_v$ di un gas: che nel caso di aria ($\gamma = 1.4$)diventa $\frac{p}{p_e} \geq 1.89$.
Anche la velocità di uscita è limitata in questo regime

$$u_e = \sqrt{\frac{2\gamma RT}{\gamma + 1}} \tag{2.11}$$

Dove T è la temperatura in Kelvin e R è la costante dei gas specifica per l'aria. Possiamo dunque scrivere la spinta come:

$$S = PA_e\gamma \left(\frac{2}{\gamma + 1}\right)^{\frac{\gamma}{\gamma+1}}. \tag{2.12}$$

## 2.4 Quarto Regime

Nel quarto regime cade la limitazione sonica all'ugello, la portata in massa e velocità non sono più limitate, ciò avviene con un rapporto di pressione $\frac{p}{p_e}$ inferiore a 1.89

$$v_e = \sqrt{\frac{2\gamma RT}{\gamma - 1}\left[1 - \left(\frac{P_e}{P}\right)^{\frac{\gamma-1}{\gamma}}\right]} \tag{2.13}$$

$$\dot{m}_a = \frac{pA_e}{\sqrt{RT}}\sqrt{\frac{2\gamma}{\gamma - 1}\left(\frac{p_g}{p}\right)^{\frac{1}{\gamma}}}\sqrt{1 - \left(\frac{p_g}{p}\right)^{\frac{\gamma-1}{\gamma}}} \tag{2.14}$$

Il che ci dà spinta fino al completo raggiungimento dell'equilibrio tra pressione interna ed esterna. Si è deciso di non modellare l'effetto della variazione della massa del razzo dovuto alla portata in massa di aria, poiché la variazione tra la massa d'aria prima del lancio e al raggiungimento dell'equilibrio è di soli $5g$, cioè meno del 2% della massa totale del razzo ($240g$)

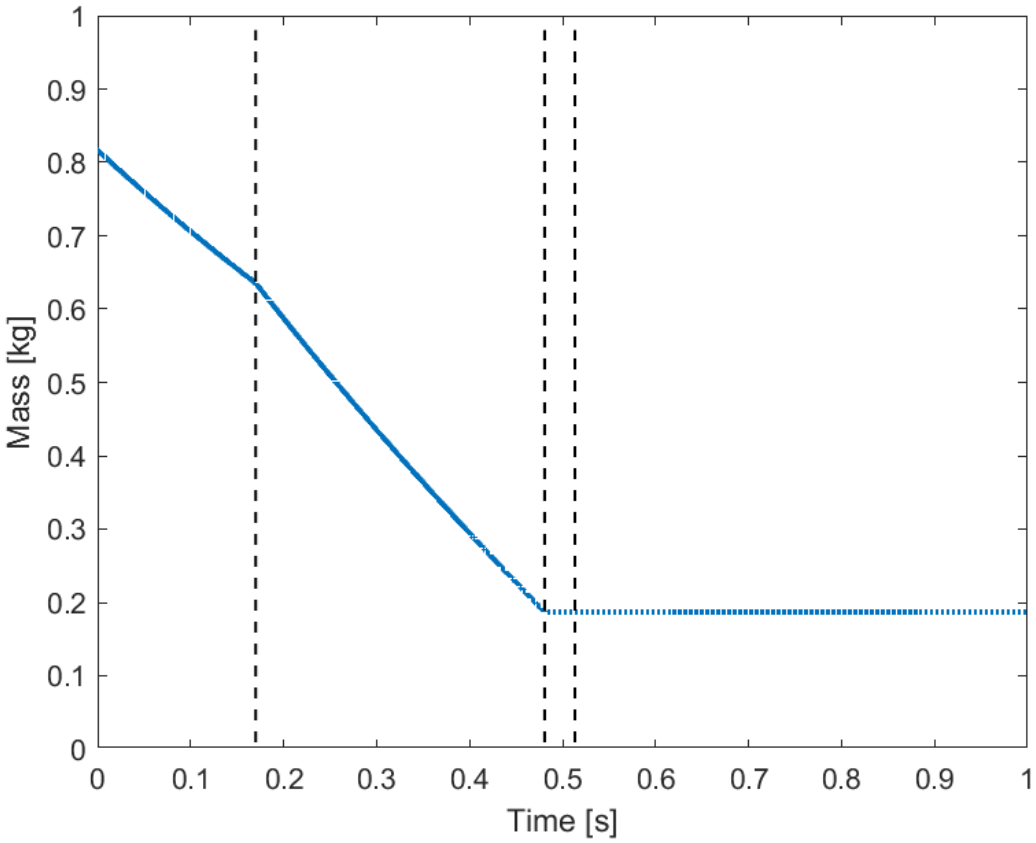



Figura 2.3: Massa nel tempo

# 3 Modello Costitutivo

Ora discuteremo come la pressione interna della bottiglia varia nel tempo e quali modelli siano applicabili per la simulazione della stessa.

## 3.1 Primo e Secondo Regime

Innanzitutto dobbiamo modellare l'espansione del gas all'interno della bottiglia, cosa che facciamo con un'espansione isoterma

$$pV = p_0V_0 = cost$$

$$\frac{d}{dt}(pV) = \frac{dp}{dt}V + p\frac{dV}{dt} = 0 \tag{3.1}$$

$$\frac{dp}{dt} = -\frac{p}{V}\frac{dV}{dt}$$

ma per l'equazione 3.1 possiamo scrivere

$$\frac{dp}{dt} = -\frac{p^2}{p_0V_0}\frac{dV}{dt}$$

Inoltre possiamo scrivere

$$\frac{dV}{dt} = A_eu_e \tag{3.2}$$

$$\frac{dp}{dt} = -\frac{p^2}{p_0V_0}A_eu_e \tag{3.3}$$

## 3.2 Terzo Regime

Per modellare la variazione di pressione nel terzo e quarto regime seguiremo le ipotesi utilizzate da Utsav[4]

$$\frac{dp}{dt} = \frac{p_e}{m_f}\frac{dm}{dt} \tag{3.4}$$

dove $m_f$ è la massa d'aria finale al raggiungimento dell'equilibrio

## 3.3 Quarto Regime

$$\frac{dp}{dt} = -\frac{p_e A_e}{L_{pc} A_s} \sqrt{\frac{2\gamma RT}{\gamma - 1} \left(1 - \left(\frac{p_e}{p}\right)^{\frac{\gamma-1}{\gamma}}\right)} \tag{3.5}$$

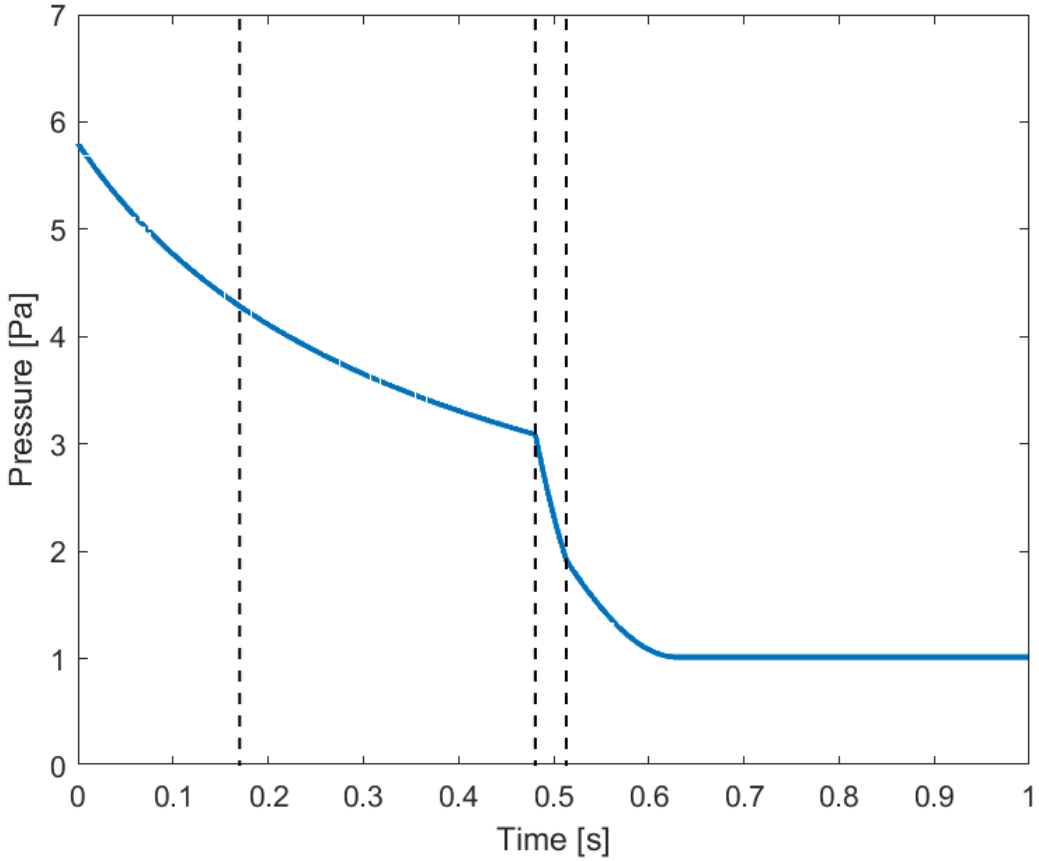



Figura 3.1: Pressione nel tempo

# 4 Unita Inerziale


Per la raccolta dei dati sperimentali del movimento del razzo si è deciso di installare all'interno dell'ogiva un'unità Inerziale (da qui in poi IMU), un sistema dotato di diversi sensori in grado di misurare le accelerazioni e velocità angolari percepite dal razzo, in modo da poterne ricavare il movimento per un breve intervallo di tempo.
Il sensore in questione è l'unità Inerziale WT901SDCL realizzata da Witmotion. I dati della stessa vengono salvati all'interno di una scheda SD che funge anche da interruttore d'accensione per l'unità stessa.
Una volta raccolti, i dati vengono salvati su SD sotto forma di *file.txt* in codice binario e devono essere quindi trasferiti su computer e convertiti. La ditta di realizzazione del sensore fornisce due possibili soluzioni:

- l'applicazione, sempre di proprietà di Witmotion che permette, sia di avere una rappresentazione grafica dei dati raccolti, che di esportarli sotto forma di *file.txt*

- un codice Python per trasformare i dati senza dover passare dall'App, codice che si è rivelato contenere degli errori nella trascrizione dei tempi e che quindi è stato necessario modificare

Nel caso di questo testo si è scelto di procedere con la seconda opzione, cioè lo script in Python modificato, che verrà collocato nell'appendice per renderlo disponibile.

## 4.1 Criticità

I sensori di questo tipo non vengono normalmente utilizzati per calcolare la posizione di un oggetto poiché, dovendo integrare due volte l'accelerazione, sono noti per un accumulo dell'errore quadratico con conseguente deriva rispetto alla posizione reale. Ciononostante, essendo l'intervallo di tempo dell'ordine dei secondi, si è deciso di procedere comunque, riscontrando nella raccolta dei dati un effettivo accumulo di errore trascurabile.
Inoltre questo particolare modello di unità Inerziale presenta un problema con la calibrazione del magnetometro che ha reso impossibile l'utilizzo dello stesso, anche se il problema non ha influenzato la raccolta e l'utilizzo dei dati, ciò rende impossibile orientare il moto del razzo rispetto al *NORD*

## 4.2 Sistemi di riferimento

Essendo sull'unità Inerziale già presenti tre assi corrispondenti a quelli delle letture del sensore stesso, si è deciso di adottare quelli come sistema di riferimento solidale al corpo *(Body)*.
In particolare l'asse $Z$ corrisponderà con l'asse longitudinale del razzo per motivi di semplicità di installazione.

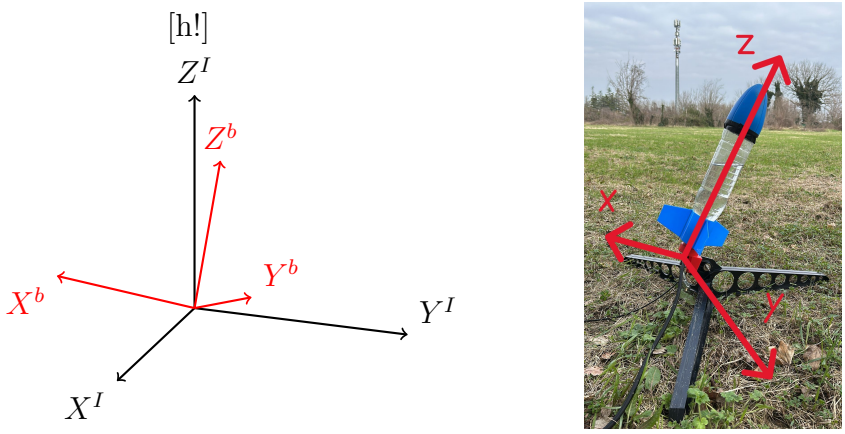



Figura 4.1: Sistema di riferimento locale

Per il sistema di riferimento Inerziale invece si è optato per un modello di **terra piatta non rotante**, approssimazione giustificata dal moto nello spazio di poche decine di metri.
In particolare, non essendo funzionante il magnetometro sull'IMU per la nostra applicazione utilizzeremo una terna Inerziale non *NED (North-East-Down)* ma semplicemente 3 assi $X$-$Y$-$Z$ con $Z$ allineato al centro della Terra e in direzione verso l'alto, mentre $X$ e $Y$ giacenti sul piano normale a $Z$.

## 4.3 Assetto

In generale si intende per assetto l'orientamento di un sistema di riferimento qualsiasi rispetto ad un altro, nel nostro caso in particolare intenderemo l'orientamento del sistema Body rispetto a quello Inerziale.
Per questa applicazione nello specifico utilizzeremo gli angoli Eulero per descrivere l'assetto, anche se è bene sottolineare che esistano anche altri modi quali ad esempio i quaternioni.

## 4.4 Angoli di Eulero

Gli angoli di Eulero sono tre angoli che permettono di definire l'orientamento di due sistemi di riferimento partendo da una situazione in cui i due sono coincidenti, effettuando delle rotazioni in una determinata e specifica sequenza, nel nostro caso utilizzeremo la terna $3-2-1$ e considereremo la rotazione antioraria come positiva.

Il primo angolo $\psi$ è noto come *Azimuth* o *angolo di prua*, esso definisce la rotazione attorno al terzo asse, cioè $Z$, ed è utilizzato per definire la rotazione dell'asse $X$ rispetto alla direzione

*NORD*. Nel nostro caso però indicherà semplicemente la rotazione rispetto alla direzione dell'asse $X$ nel momento dell'accensione del sensore.

Partendo dalla nuova posizione raggiunta, il secondo angolo $\theta$ è noto come *angolo di elevazione*, esso definisce la rotazione attorno al secondo asse, $Y$.

Infine l'ultimo angolo $\phi$ detto *angolo di inclinazione* definisce la rotazione attorno al primo asse $X$
Questi angoli possono essere ottenuti integrando le equazioni di propagazione degli angoli di Eulero.

$$\begin{cases} \dot{\phi} = \omega_x + \omega_y \sin\phi \tan\theta + \omega_z \cos\phi \tan\theta \\ \dot{\theta} = \omega_y \cos\phi - \omega_z \sin\phi \\ \dot{\psi} = \frac{\omega_y \sin\phi + \omega_z \cos\phi}{\cos\theta} \end{cases} \tag{4.1}$$

Dove $\omega = [\omega_x, \omega_y, \omega_z]$ rappresenta la velocità di rotazione del sistema Body rispetto a quello Inerziale.
Ciononostante, essendo l'applicazione per cui questo modello di IMU proprio quella di stimare gli angoli $\psi$, $\theta$ e $\phi$, si è deciso di utilizzare proprio le stime dell'unità Inerziale per ottenere risultati più precisi, includendo esse diversi tipi di filtri che vanno oltre lo scopo di questa tesi.

## 4.5 Modello Matematico

Le accelerazioni misurate dall'unità Inerziale possono essere descritte con il seguente modello[5]

$$a = a_m + g - b$$

Dove $a$ è l'accelerazione rispetto ad un sistema di riferimento Inerziale, $g$ quella gravitazionale, $a_m$ l'accelerazione percepita dal sensore e $b$ il bias che per il momento ignoreremo.

$$a^I = C^{Iner}_{Body}(a^b_m) + g^I \tag{4.2}$$

Dove $a^I$ è l'accelerazione reale in assi Inerziali, $a^b_m$ accelerazione misurata in assi del sistema di riferimento locale, $g^I$ accelerazione gravitazionale in assi Inerziali e $C^I_b$ la matrice di rotazione tra sistema di riferimento solidale (Body) e quello Inerziale. Ciò quindi significa che per poter operare con le accelerazioni rispetto al sistema di riferimento Inerziale è necessario trovare prima la matrice di rotazione da Body a Inerziale, che inoltre coincide con l'inversa della matrice di rotazione da Inerziale a Body.

$$C^{Iner}_{Body} = C^{Body^T}_{Iner}$$

## 4.6 Matrice di Rotazione

La matrice di rotazione tra due sistemi di riferimento può essere trovata facendo la composizione delle tre matrici che descrivono le singole rotazioni attorno agli assi, quindi nel caso della rotazione da sistema di riferimento Inerziale a Body $C_{Iner}^{Body}$ può essere scritta come:

$$\mathbf{C}_{\mathbf{Iner}}^{\mathbf{Body}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{C}_{\mathbf{Iner}}^{\mathbf{Body}} = \begin{bmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ \sin\theta\sin\phi\cos\psi - \cos\phi\sin\psi & \sin\theta\sin\phi\sin\psi + \cos\phi\cos\psi & \sin\phi\cos\theta \\ \sin\theta\cos\phi\cos\psi + \sin\phi\sin\psi & \sin\theta\cos\phi\sin\psi - \sin\phi\cos\psi & \cos\phi\cos\theta \end{bmatrix} \tag{4.3}$$

Quindi possiamo scrivere la matrice di rotazione da Body a Inerziale, inversa di quella appena trovata, come:

$$C_{Body}^{Iner} = C_{Iner}^{Body^T}$$

$$\mathbf{C}_{\mathbf{Body}}^{\mathbf{Iner}} = \begin{bmatrix} \cos\theta\cos\psi & \sin\theta\sin\phi\cos\psi - \cos\phi\sin\psi & \sin\theta\cos\phi\cos\psi + \sin\phi\sin\psi \\ \cos\theta\sin\psi & \sin\theta\sin\phi\sin\psi + \cos\phi\cos\psi & \sin\theta\cos\phi\sin\psi - \sin\phi\cos\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix} \tag{4.4}$$

## 4.7 Breve verifica dei dati

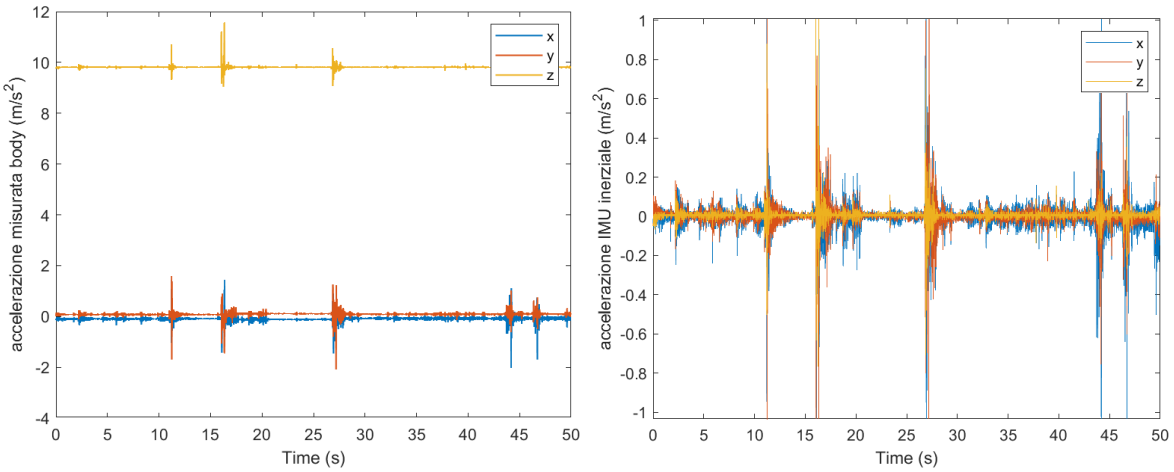



Figura 4.2: Confronto accelerazioni body e inerziale

Ad esempio nella figura sopra abbiamo la raccolta dati di un test statico, ignorando i picchi dovuti a movimenti bruschi, si nota come si è quasi perfettamente riusciti a portare le accelerazioni da terna Body a Inerziale in quanto il corpo risulta pressoché fermo.

## 4.8 Lettura dei dati

Come già accennato precedentemente, l'IMU raccoglie i dati sotto forma di *file.txt* codificati. È stato quindi necessario adattare uno script fornito dal distributore per decodificare i file, sempre sotto forma di *.txt* e successivamente scrivere una funzione Matlab che permettesse di leggere i dati su MATLAB.
Per isolare le letture dei dati alla sola fase di volo si è proceduto ad un'analisi dell'andamento dell'accelerazione (indifferentemente Body o Inerziale) di ogni singola registrazione.
Nel momento del decollo si passa chiaramente da una condizione di stazionarietà a una di moto distinguibile per l'elevata variazione di accelerazione; successivamente, nel momento di contatto con il terreno, si ha una seconda chiara e brusca variazione della stessa.

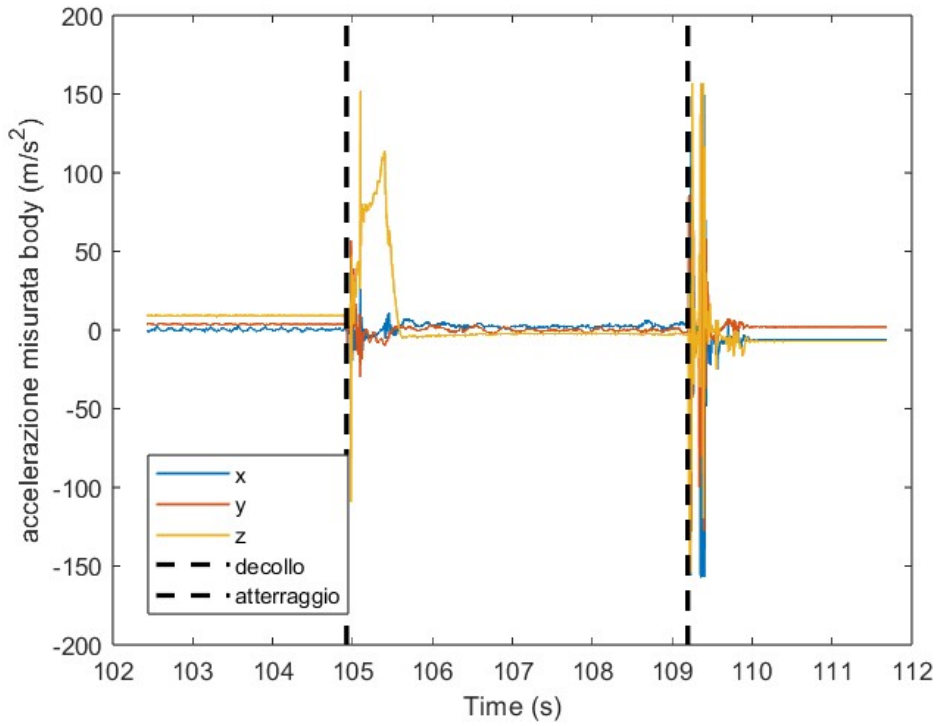



Figura 4.3: Intervallo di tempo di un lancio

Ad esempio in questo caso il volo dura da $104.9250s$ a $109.1950s$

## 4.9 Manipolazione dati

Una volta che le accelerazioni sono state ruotate nel sistema di riferimento Inerziale ed a esse è stato sottratto il vettore gravità $[0, 0, g]$ è stato sufficiente integrarle per ottenere la velocità nel tempo e, integrando nuovamente, la posizione.

$$a^I(t) = C^{Iner}_{Body}(t)(a^b_m(t)) + g^I$$

Si noti come $C_{Body}^{Iner}(t)$ viene ottenuta con l'equazione 4.4

$$s^I(t) - s^I(t_i) = \int_{t_i}^{t}\int_{t_i}^{\tau} \left(a^I(s)\right) dsd\tau = \int_{t_i}^{t} \left(v^I(\tau)\right) d\tau$$

```
%% calcolo accel Inerziale

a_m_i=zeros(size(a_m_b));

for i=1:n
      a_m_i(i,:)=(a_m_b(i,:))*RotM(:,:,i);
      a_i(i,:)=a_m_i(i,:)-[0,0,9.81];
end
%% integrazione

v_i=cumtrapz(time,a_i);

s_i=cumtrapz(time,v_i);
```

## 4.10 Bias

Da diversi campionamenti è risultato che spesso i voli sembravano terminare a mezz'aria a qualche metro da terra, il momento di atterraggio risulta sempre chiaramente distinguibile grazie all'improvvisa discontinuità delle accelerazioni.
Sapendo a priori della possibilità della presenza di bias costanti nella raccolta dati delle accelerazioni e che questi ultimi subiscano un doppio processo di integrazione, ci aspettiamo una divergenza di tipo parabolico tra la posizione reale e quella calcolata dall'IMU.

Nel caso specifico della coordinata $Z$ sappiamo che il razzo dovrà necessariamente atterrare alla stessa quota del decollo, possiamo quindi trovare il bias nella sua componente verticale.

$$a(t) = a_m(t) + g - b = a_c(t) - b$$

dove con $a_c(t)$ andiamo a considerare sia l'accelerazione misurata che quella gravitazionale, solo a scopo di calcolo.

$$v(t) - v(t_i) = \int_{t_i}^{t} a(\tau)d\tau = \int_{t_i}^{t} \left(a_c(\tau) - b\right) d\tau = \int_{t_i}^{\tau} a_c(\tau)d\tau - b(t - t_i)$$

$$s(t) - s(t_i) = \int_{t_i}^{t}\int_{t_i}^{\tau} a(s)dsd\tau = \int_{t_i}^{t}\int_{t_i}^{\tau} \left(a_c(s) - b\right) dsd\tau$$

$$s(t) - s(t_i) = \int_{t_i}^{t}\int_{t_i}^{\tau} \left(a_c(s)\right) dsd\tau - \int_{t_i}^{t}\int_{t_i}^{\tau} bdsd\tau = \int_{t_i}^{t}\int_{t_i}^{\tau} \left(a_c(s)\right) dsd\tau - \frac{b(t-t_i)^2}{2} \qquad (4.5)$$

Ma visto che sappiamo che al tempo $s_z(t = t_f) - s_z(t_i) = 0$

$$s_z(t_f) - s_Z(t_i) = \int_{t_i}^{t} \int_{t_i}^{\tau} (a_{c,z}(s))\, ds d\tau - \frac{b_z(t - t_i)^2}{2} = 0$$

$$b_z = \frac{2 \int_{t_i}^{t} \int_{t_i}^{\tau} (a_{c,z}(s))\, ds d\tau}{(t - t_i)^2} \tag{4.6}$$

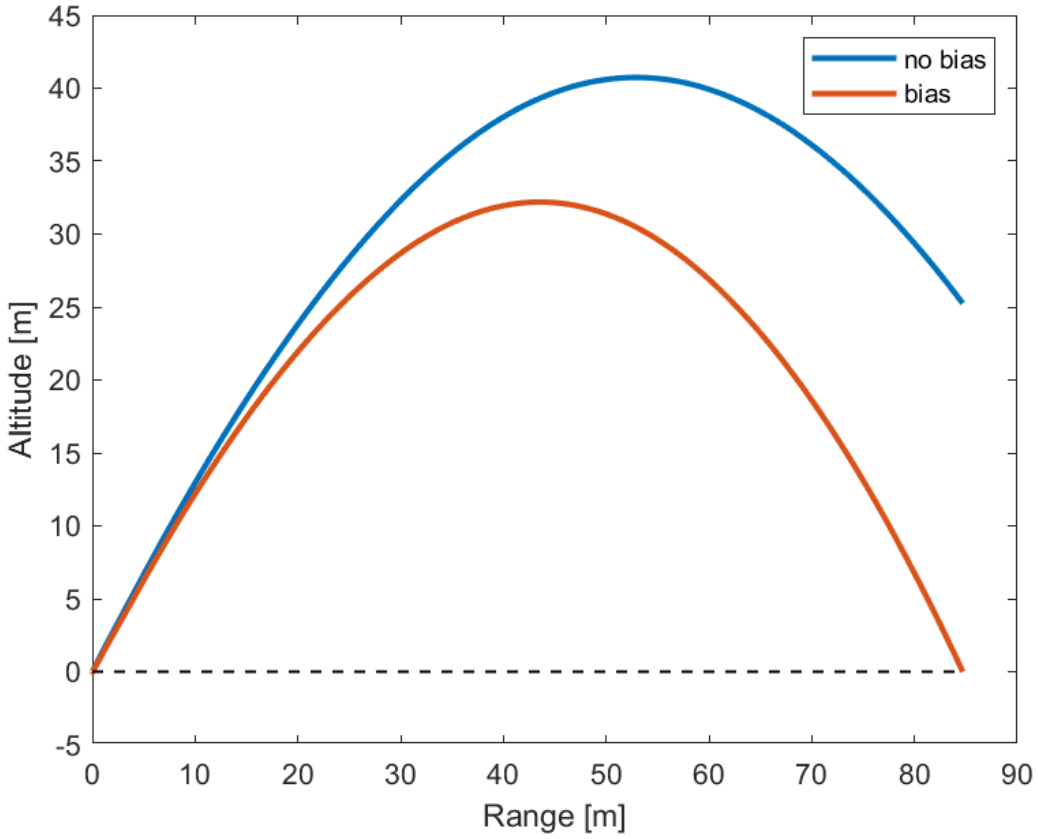
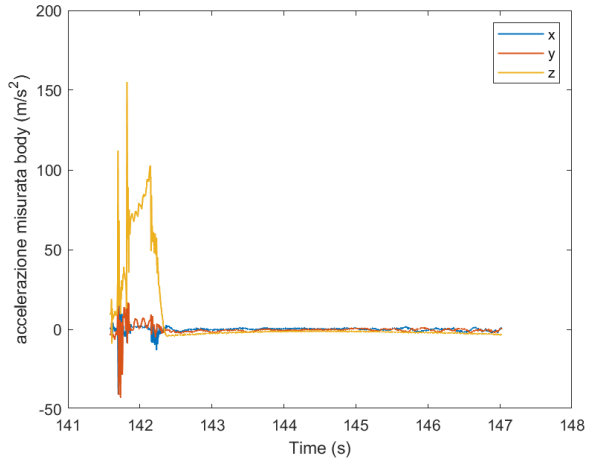


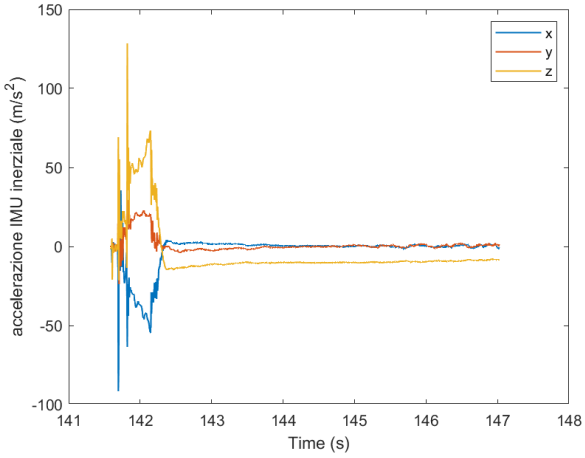

Figura 4.4: Confronto Bias

## 4.11 Integrazione volo completo

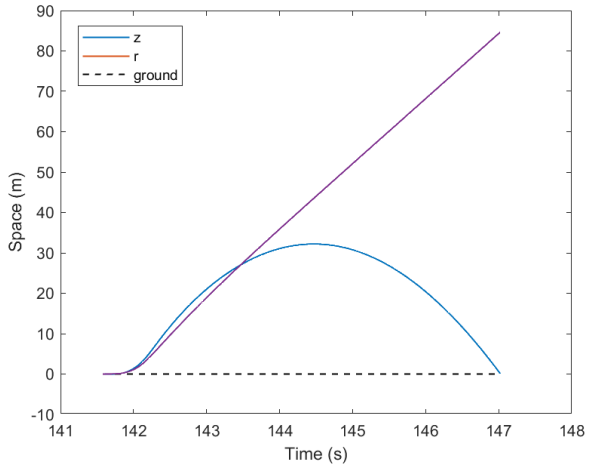
Considerando tutte le nozioni esposte sopra, è stato creato un codice in ambiente Matlab per analizzare i dati raccolti; i dati che seguono sono la graficazione dei campioni raccolti a seguito del primo volo effettuato con successo.



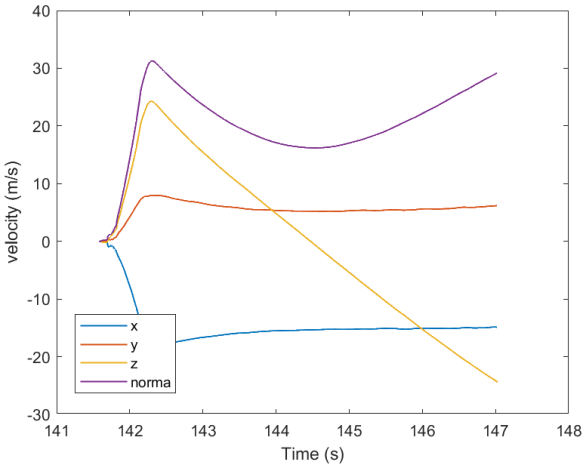

(a) Accel. Body



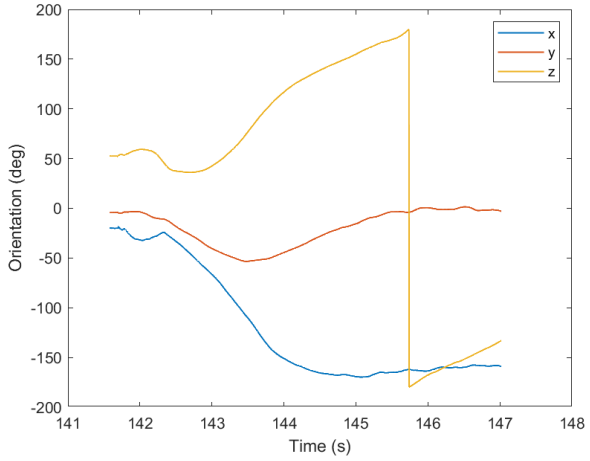

(b) Accel. Inerz



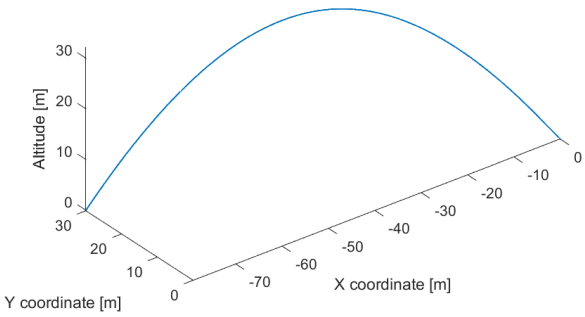

(c) Spazio-Tempo




(d) Velocità-Tempo




(e) Assetto-Tempo




(f) Traiettoria 3D

# 5 Simulazione Numerica

## 5.1 Ricerca Bibliografica

Si è partiti con lo studio teorico delle leggi che governano il moto bidimensionale di un razzo sul piano x-z mediante lo studio bibliografico in letteratura scientifica[2]

## 5.2 Dinamica e Cinematica

Si è deciso di realizzare un modello di punto materiale, sono state considerate alcune ipotesi semplificative:

- angolo di attacco $\alpha = 0$, con conseguente annullamento delle forze aerodinamiche laterali e un allineamento tra la velocità e la spinta del razzo
- il moto avviene su un terra piatta non rotante
- assenza di vento

le equazioni che quindi possono descrivere la dinamica e la cinematica del razzo in un piano x-z sono:

$$\begin{cases} \dot{v} = \frac{S-D}{m}\cos(\alpha) - g\sin(\gamma) \\ \dot{\gamma} = \frac{S-D}{m}\sin(\alpha) + (-\frac{g}{v})\cos(\gamma) \\ \dot{z} = v\sin(\gamma) \end{cases}$$

Dove $S, D, \gamma, v$ sono rispettivamente Spinta, Resistenza, angolo di pendenza della traiettoria. Le equazioni sono state manipolate per adattarle al nostro modello aggiungendo la seconda equazione della cinematica considerando un moto piano per poter studiare il moto orizzontale oltre a quello verticale

$$\begin{cases} \dot{v} = \frac{T}{m} - g\sin(\gamma) \\ \dot{\gamma} = (-\frac{g}{v})\cos(\gamma) \\ \dot{z} = v\sin(\gamma) \\ \dot{x} = v\cos\gamma \end{cases} \tag{5.1}$$

## 5.3 Volo $\gamma$ bloccato

Durante il volo, nei primi istanti dal lancio, la spina utilizzata per mandare in pressione la bottiglia resta ancora inserita all'interno della bottiglia, imponendo un $\gamma$ fisso. L'argomento

viene meglio affrontato nel capitolo sulla Propulsione, teniamo comunque conto di questa ipotesi anche nella modellazione della dinamica.

$$\begin{cases} \dot{v} = \frac{T}{m} - g\sin(\gamma) & \\ \dot{\gamma} = 0 & se\ t < t_{libero} \\ \dot{\gamma} = (-\frac{g}{v})\cos(\gamma) & se\ t > t_{libero} \\ \dot{z} = v\sin(\gamma) & \\ \dot{x} = v\cos\gamma & \end{cases} \tag{5.2}$$

## 5.4 Modellazione Resistenza aerodinamica

Per modellare la resistenza aerodinamica si è semplicemente utilizzata la formula

$$D = \frac{1}{2}\rho A_s v^2 c_D \tag{5.3}$$

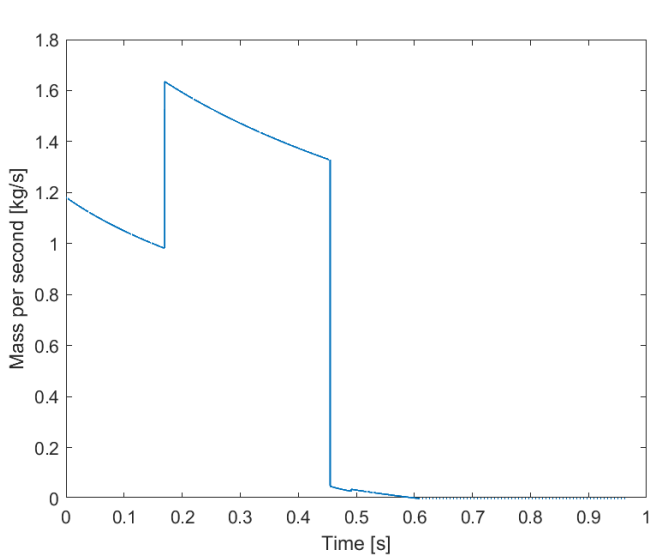
Dove $A_s, c_D, \rho, v$ sono rispettivamente la sezione del razzo, il coefficiente di resistenza aerodinamica, densità dell'aria e velocità del razzo.
Per la stima del $c_d$ si è preliminarmente utilizzato un valore di 0.345 come calcolato da Barrio-Perotti[1] in attesa che ulteriori studi su questo particolare modello di razzo ad acqua siano pubblicati.

## 5.5 Modellazione della spinta

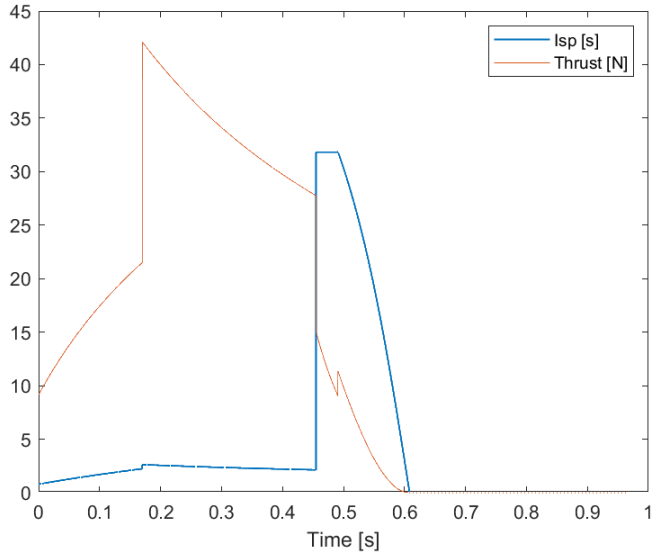
Per modellare la spinta si è utilizzato un blocco Simulink implementando le equazioni del capitolo sul Modello Propulsivo, in particolare l'output del modello sono la portata in massa $\dot{m}$ e velocità di uscita del flusso $u_e$.

$$S = \dot{m}u_e = \dot{m}Ispg_0$$

Possiamo quindi trovare l'$ISP, u_e, \dot{m}$ e $S$ per il nostro razzo ad acqua istante per istante.




(a) Portata in massa




(b) Thrust e Isp

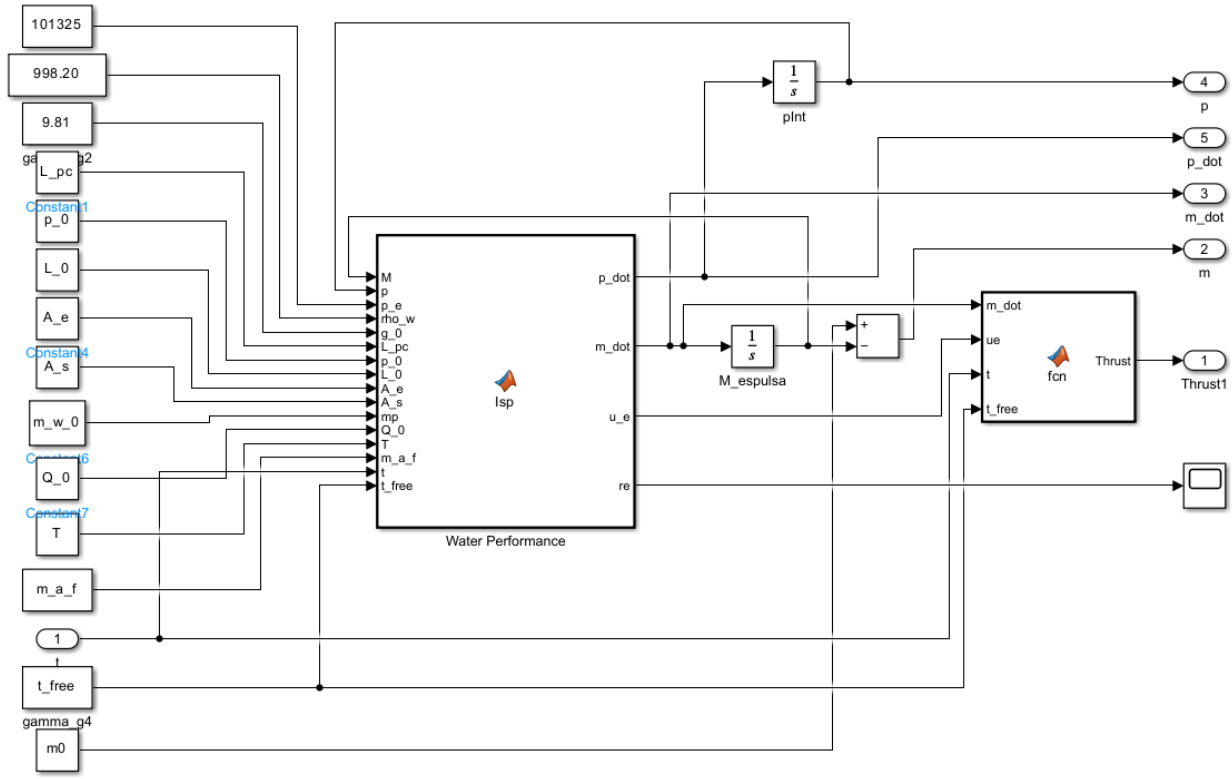



Figura 5.1: Sottosistema Propulsivo

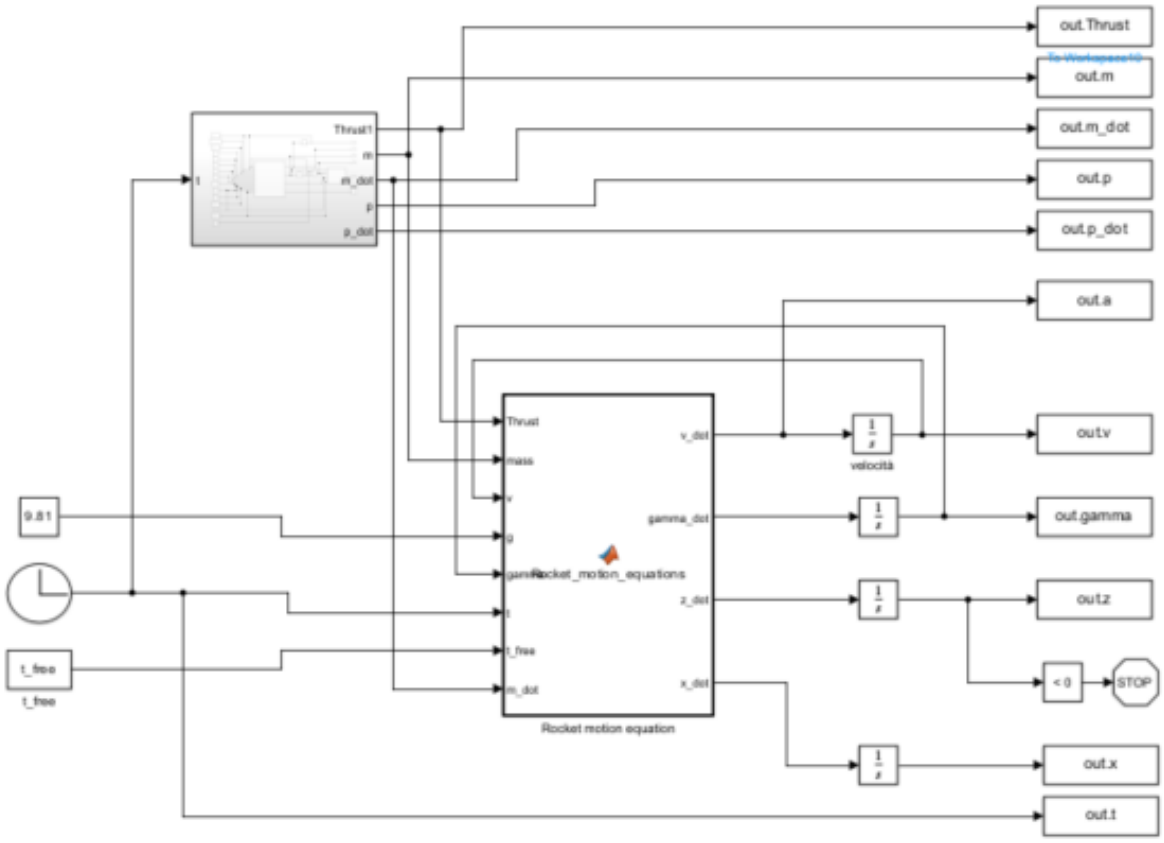



Figura 5.2: Sistema Simulink

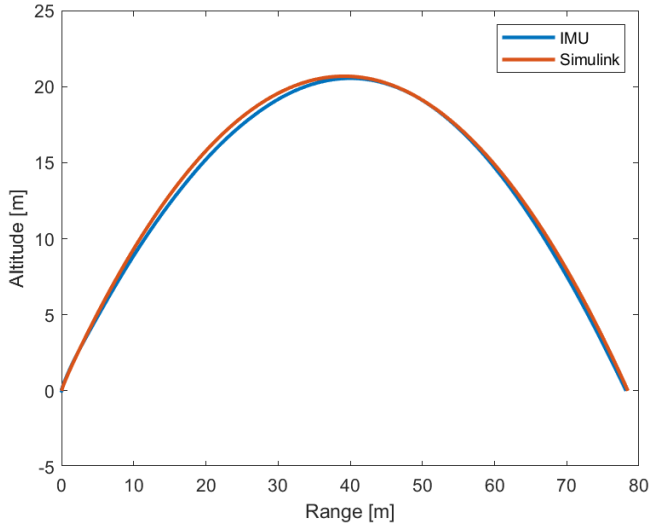
## 5.6 Inizializzazione

Qui di seguito si presentano i valori che sono stati utilizzati per la simulazione del razzo ad acqua.

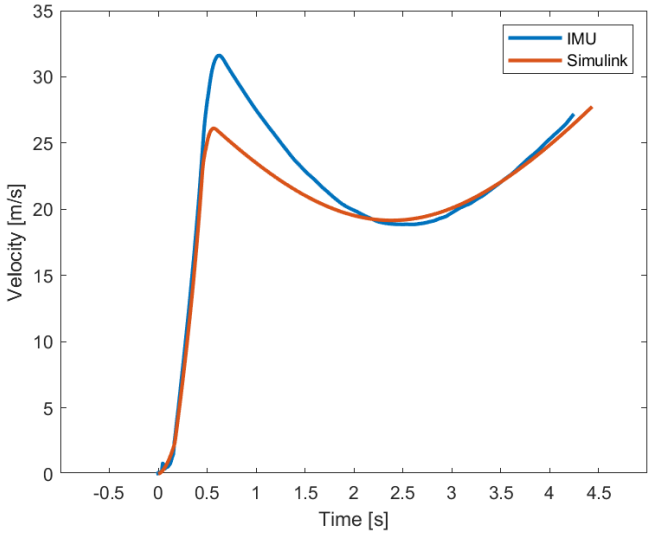
```
%% inizializzazione

p_0=580000; %[Pa] pressione iniziale
L_pc=0.21; %[m] lunghezza camera di pressione
m_w_0=0.600; %[kg] massa d'acqua
t_free=0.170;

nozzle_r=0.0045; %[m] raggio di gola uggello
bottle_r=0.0475; %[m] raggio bottiglia
me=0.240;    %[kg] massa a secco
T=291.15; %[K] temperatura
R=287; %[J/kg-K] Specific gas constant

%% condizioni iniziali

v0=0; %[m/s]
gamma0_deg=57; %[deg]
z0=0; %[m]
x0=0; %[m]

%% derivazione grandezze

rho_w=998.20;
gamma0=gamma0_deg*(pi/180);
m0=me+m_w_0;   %[kg] massa totale
A_e=nozzle_r^2*pi; %[m^2] area di gola uggello
A_s=bottle_r^2*pi; %[m^2] area della bottiglia
V_w_0=m_w_0/rho_w; %volume acqua iniziale
Q_0=A_s*L_pc-V_w_0; %volume aria iniziale
L_0=V_w_0/A_s; %livello acqua iniziale
m_a_0=p_0*Q_0/(R*T); %[kg] massa d'aria iniziale
m_a_f=101325*Q_0/(R*T); %[kg] massa d'aria finale
```

# 6 Risultati e Confronto

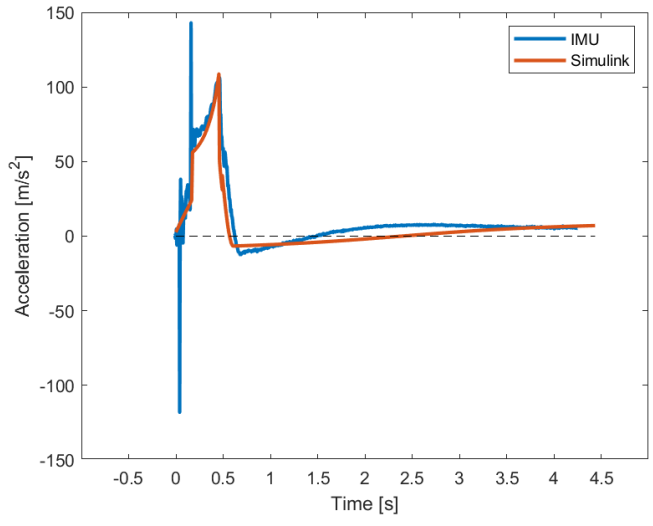
Vediamo ora di confrontare i risultati di un lancio e della sua parallela simulazione numerica. In particolare analizziamo la traiettoria, oltre che velocità, accelerazione e posizione nel tempo.



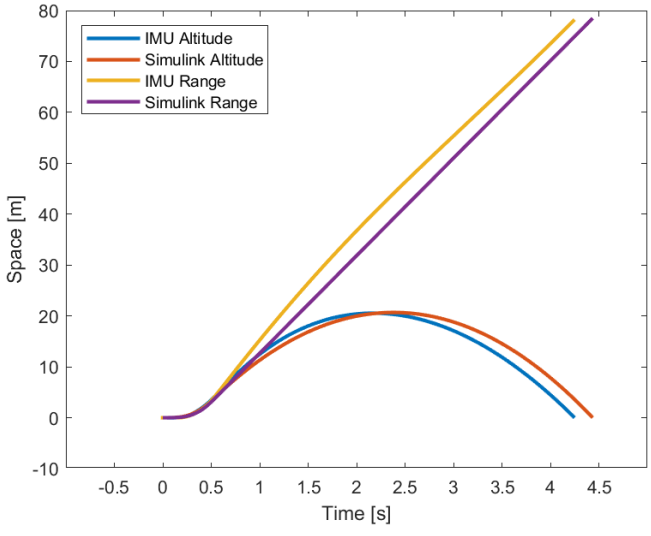

(a) Traiettoria




(b) Velocità-Tempo




(c) Accel.-Tempo




(d) Spazio-Tempo

Come si può vedere i risultati raggiunti sono piuttosto soddisfacenti, con una discreta similitudine tra modello teorico e dati sperimentali.
La discrepanza maggiore si ha tra le velocità massime raggiunte dal razzo, ciononostante il tempo di volo risulta pressoché identico, evidenziando la validità del modello.
I profili delle accelerazioni al decollo si presentano quasi identici, ciò era comunque atteso poiché

il primo regime è stato modellato su dati reali, in ogni caso anche gli altri regimi danno risultati più che soddisfacenti sia in accelerazione che in durata dello stesso.

# 7 Conclusioni

Il presente testo ha permesso di sviluppare un modello per la simulazione del volo in un lanciatore in scala quale razzo ad acqua, unendo la modellazione teorica della propulsione con l'implementazione delle equazioni della dinamica e cinematica per punto materiale. L'utilizzo di Simulink ha permesso di implementare questo modello istante per istante in maniera accurata e intuitiva.
Fondamentale è stato lo sviluppo di un metodo per utilizzare l'unità inerziale, con il risultato secondario di essere riusciti ad aggirare la necessità di utilizzare l'applicazione proprietaria di Witmotion, nonostante lo script fornito fosse errato.
Lo sviluppo di questi codici fornisce la base per possibili sviluppi futuri in successivi progetti di ricerca e implementazioni didattiche.

## 7.1 Limitazioni

Le maggiori limitazioni di questo progetto sono state, la difficoltà nel determinare un reale angolo di decollo $\gamma_0$, dove sono state fatte ipotesi piuttosto forti .
Non è stato possibile conoscere i valori reali di portata in massa durante il volo e dello sviluppo della pressione durante il volo, e anche nel caso della pressione iniziale le misure raccolte da terra prima del lancio sono state limitate dalla strumentazione disponibile (pompa di bicicletta).
E' inoltre auspicabile una verifica sperimentale del valore del $c_D$.
Non è chiaro se il problema al magnetometro dell'unità inerziale sia risolvibile con questo modello e sia necessario per applicazioni future che richiedono la conoscenza del NORD magnetico l'utilizzo di un'altra unità.

# A Codice Dinamica e Cinematica

```
1  function [v_dot,gamma_dot,z_dot,x_dot] = Rocket_motion_equations(
       Thrust,mass,v,g,gamma,t,t_free,m_dot)
2
3
4  v_dot=Thrust/mass-g*sin(gamma)-(.5*1.245*6.36e-05*v^2*0.345)/mass;%-
       m_dot*v/(mass);
5
6  z_dot=v*sin(gamma);
7  x_dot=v*cos(gamma);
8
9  if t<t_free
10
11     gamma_dot = 0;
12 else
13     % gamma_dot=(x_dot*v_z_dot-z_dot*v_x_dot)/(z_dot^2+x_dot^2);
14
15     gamma_dot=-(g/v)*cos(gamma);
16 end
17
18
19 end
```

# B Codice Propulsione

```
1  function [p_dot,m_dot,u_e,re] = WaterPerformance(M,p,p_e,rho_w,g_0,
       L_pc,p_0,L_0,A_e,A_s,mp,Q_0,T,m_a_f,t,t_free)
2
3  R=287; %[J/kg-K] Specific gas constant
4  gamma=1.4; %[] Ratio of specific heat
5
6  if t<t_free
7
8      re=-1;
9      u_e=( ( 2*(p-p_e)/rho_w + 2*g_0* (L_pc-p_0*(L_pc-L_0)/p) )/( 1-(
           A_e/A_s)^2) )^.5;
10     p_dot=-(p^2/(p_0*Q_0))*A_e*u_e;
11     m_dot_w=rho_w*A_e*u_e*3/5;
12     m_dot_a=0;
13
14 elseif M<mp
15
16     re=0;
17     u_e=( ( 2*(p-p_e)/rho_w + 2*g_0* (L_pc-p_0*(L_pc-L_0)/p) )/( 1-(
           A_e/A_s)^2) )^.5;
18     p_dot=-(p^2/(p_0*Q_0))*A_e*u_e;
19     m_dot_w=rho_w*A_e*u_e;
20     m_dot_a=0;
21
22
23 elseif p/p_e > (2 / (gamma + 1))^(-gamma / (gamma - 1))
24
25     re=1;
26     m_dot_w=0;
27     m_dot_a=p*A_e*sqrt(gamma*(2/(gamma+1))^((gamma+1)/(gamma-1))/(R*
           T));
28     u_e=sqrt(2*gamma*R*T/(gamma+1));
29     p_dot=-p_e*m_dot_a/m_a_f;
30
31 elseif p>p_e && p/p_e<(2 / (gamma + 1))^(-gamma / (gamma - 1))
32
33     re=2;
34     m_dot_w=0;
35     u_e = sqrt((2 * gamma * R * T / (gamma - 1)) * (1 - (p_e / p)^((
           gamma - 1) / gamma)));
```

```
36      p_dot=-(p_e*A_e/(L_pc*A_s)*sqrt(2*gamma*R*T/(gamma-1)*(1-(p_e/p)
            ^((gamma-1)/gamma))));
37      % m_dot_a=p*u_e*A_e/(R*T);
38      pg = p_e;
39      m_dot_a = (p * A_e ...
40          / sqrt(R * T)) * ...
41          sqrt((2 * gamma / (gamma - 1)) * (pg / p)^(1/gamma)) * ...
42          sqrt(1 - (pg / p)^((gamma - 1)/gamma));
43  else
44
45      re=3;
46      m_dot_w=0;
47      u_e=0;
48      p_dot=0;
49      m_dot_a=0;
50
51  end
52
53  m_dot=m_dot_a+m_dot_w;
```

# C Codice Python Unità Inerziale

```
1  import threading
2
3  tempBuffer = []
4  data = []
5  line = ""
6
7  def read_file(filepath, new_filepath):
8      try:
9          with open(filepath, 'rb') as f:
10             while True:
11                 chunk = f.read(102400)
12                 if not chunk:
13                     print("Reading file completed")
14                     break
15                 processData(chunk, new_filepath)
16     except FileNotFoundError:
17         print("File not found:", filepath)
18     except Exception as e:
19         print("An error occurred:", e)
20
21 def write_lines_to_file(filename, lines):
22     try:
23         with open(filename, 'a') as f:
24             for line in lines:
25                 f.write(line + '\n')
26     except Exception as e:
27         print("An error occurred while writing to file:", e)
28
29 def processData(chunk, new_filepath):
30     global tempBuffer, data
31     tempdata = bytes.fromhex(chunk.hex())
32     for val in tempdata:
33         tempBuffer.append(val)
34         if tempBuffer[0] != 0x55:
35             del tempBuffer[0]
36         if len(tempBuffer) == 2 and (tempBuffer[1] & 0xf0) != 0x50:
37             del tempBuffer[0]
38         if len(tempBuffer) >= 11:
39             processPack(tempBuffer[:11])
40             del tempBuffer[:11]
41
```

```
42      if data:
43          write_lines_to_file(new_filepath, data)
44          data.clear()
45
46  def processPack(pack):
47      global data, line
48      if len(pack) < 11:
49          return
50
51      if pack[1] == 0x50:
52          if line:
53              data.append(line)
54          year = "20" + str(pack[2])
55          line = f"{year}-{pack[3]:02d}-{pack[4]:02d} {pack[5]:02d}:{
                pack[6]:02d}:{pack[7]:02d}.{pack[9] << 8 | pack[8]:03d}\t
                "
56      elif pack[1] in [0x51, 0x52, 0x53, 0x54]:
57          values = [getSignInt16(pack[i + 1] << 8 | pack[i]) for i in
                range(2, 8, 2)]
58          scales = {0x51: 16, 0x52: 2000, 0x53: 180, 0x54: 120}
59          line += "\t".join([str(round(v / 32768 * scales[pack[1]], 3)
                ) for v in values]) + "\t"
60
61  def getSignInt16(num):
62      return num - 0x10000 if num >= 0x8000 else num
63
64  if __name__ == "__main__":
65      while True:
66          filepath = input("Enter the path to the file: ")
67          print("Processing file, please wait...")
68
69          if '.' in filepath:
70              components = filepath.rsplit('.', 1)
71              new_filepath = f"{components[0]}_new.txt"
72          else:
73              new_filepath = filepath + "_new.txt"
74
75          write_lines_to_file(new_filepath, ["Time\tAccX\tAccY\tAccZ\
                tGx\tGy\tGz\tAngX\tAngY\tAngZ\tHx\tHy\tHz"])
76
77          thread = threading.Thread(target=read_file, args=(filepath,
                new_filepath))
78          thread.start()
79          thread.join()
80
81          print("File output:", new_filepath)
82          user_input = input("Enter 1 to repeat, 2 to exit: ")
83          if user_input == "2":
84              break
```

# D Codice lettura *.txt* Witmotion

```
function RawData=read_txt_witmotion(txt)


ds=tabularTextDatastore(txt,"TreatAsMissing","NA","MissingValue",0);
ds.ReadVariableNames = false;
A=readall(ds);
Numerical_Values=table2array(A(5:height(A),2:width(A)-1));
Time=table2array(A(5:height(A),1));

date_time=datetime(Time, 'InputFormat', 'yyyy-MM-dd-HH:mm:ss.SSS');
duration=date_time-date_time(1);
duration.Format = 'hh:mm:ss.SSSS';

% t=duration;
t=seconds(duration);

RawData=[t,Numerical_Values];

end
```

# E Foto del razzo ad acqua

# Ringraziamenti

Ci tengo a ringraziare tutte le persone che mi sono state vicine in questi anni, che li hanno resi senza dubbio i migliori della mia vita fino ad adesso.

- Innanzitutto,la mia famiglia, che mi ha insegnato il valore dell'istruzione, dandomi una possibilità di una vita migliore, ed è anche grazie a loro che sono riuscito a coglierla.

- Dopo di che, anche in ordine cronologico, abbiamo i ragazzi del liceo e delle medie di Bologna, abbiamo passato tanti e bei momenti insieme, dalla prima adolescenza al COVID siete sempre stati una certezza, come delle colonne portanti su cui ho potuto costruire il mio futuro.

- Dopo sono venuti i ragazzi di Forlì, voi che venivate da tutta Italia con i miei stessi sogni e speranze, alcuni sono riusciti a farcela, qualcuno lo abbiamo perso lungo la strada, altri stanno ancora combattendo; ma io voglio ringraziarvi tutti, presenti e noi, perché senza di voi oggi non sarei qua ed è stato un onore avervi al mio fianco.

- Ed infine ai ragazzi di Rimini, che ho conosciuto così dannatamente tardi, ma che sono riusciti a darmi così tanto in così poco tempo. Dove saremmo se Tobia non ci avesse presentati?

Alla fine ce l'abbiamo fatta, siamo riusciti a far credere al mondo di meritare questo pezzo di carta, ci sono cascati.

# Bibliografia


[1] R Barrio-Perotti, E Blanco-Marigorta, K Argüelles-Díaz, and J Fernández-Oro. Experimental evaluation of the drag coefficient of water rockets by a simple free-fall test. *European Journal of Physics*, 30(5):1039, jul 2009.

[2] T. S. Feeley and J. L. Speyer. Approximate optimal guidance for the advanced launch system. Technical report, NASA, 1993.

[3] Nir Shaviv. Water propelled rocket. *ScienceBits*, 2015. Testo disponibile al sito: http://www.sciencebits.com/RocketEqs (ultima visita marzo 2025).

[4] KC Utsav, Rafat Alnouti, and Mohamad Muflehi. Prediction of water rocket performance. In *2020 Advances in Science and Engineering Technology International Conferences (ASET)*, pages 1–5, 2020.

[5] Matteo Zanzi. Elaborazione dati per la navigazione. Dispense del corso, 2024.